מדריך למשתמש ולבודק

במסמך זה נסביר כיצד בדיוק ניתן להריץ את הקוד שלנו.

ראשית, יש לוודא שכל הספריות המפורטות בקובץ libs.txt מותקנות. כעת נפרט על החלקים בקוד הרלוונטיים למשתמש:

<u>עיבוד הנתונים</u>

הקובץ המקורי שעבדנו עימו הוא המאגר Mutual Fund שזמין בקאגל (קישור בדו"ח הפרויקט). המחברת basic_data_processing.ipynb אחראית על עיבוד הקובץ הנ"ל, ושמירה של קובץ csv כפי שצריך להיכלל בסימולטור.

ניתן להריץ את כל המחברת בצורה סדרתית מהתא הראשון עד האחרון. המחברת מצפה למצוא את הקובץ Mutual Funds.csv באותה תיקייה. שלושת התאים האחרונים במחברת הם גרפים שנועדו לדו"ח, אך התא שלפניהם כולל ייצוא של הנתונים המעובדים לקובץ csv בשם funds_after_processing.csv.

לאחר עיבוד הנתונים, את הקובץ funds_after_processing.csv יש לשים בתיקייה הראשית של הפרויקט.

הרצה מחדש של המחברת הנ"ל תייצר קובץ שיהיה מעט שונה מהקובץ שאנחנו השתמשנו בו לאור העובדה שישנו שימוש בהגרלה אקראית של רעש. את הקובץ שאנחנו עבדנו עימו לא צירפנו לתיקיית הפרויקט לאור גודלו (260Mb), אך ניתן למצוא אותו בגוגל דרייב (קישור בסוף המסמך).

<u>הרצת הסימולטור</u>

הפירוט הנ"ל מניח שהקובץ funds_after_processing.csv נמצא בתיקייה הראשית של הפרויקט, אחרת יש לשנות את הנתיב בהתאם.

נתאר כיצד להריץ ניסוי בודד עם משקיע מסוים ולאחר מכן נתאר כיצד להריץ מס' ניסויים רב בבת אחת ולתעד את התוצאות שלהם לטובת ניתוח.

תחילה יש לבחור סוג משקיע מבין המשקיעים השונים הנמצאים בתיקיית investors_types. כדי להריץ את הסימולטור בהרצה בודדת, יש לתת את המשקיע המבוקש כפרמטר לפונקציה run_simulator_by_investor בקובץ simulator.py.

לדוגמה: כדי להריץ את הסימולטור עם משקיע מסוג BestReturnLastYearInvestor יש להריץ את השורה הבאה:

```
run_simulator_by_investor(investor_type=BestReturnLastYearInvestor)
```

(שימו לב שהפרמטר investor_type מקבל את ה-type של המשקיע ולא אובייקט ספציפי – כלומר אין צורך ליצור instance חדש של האובייקט)

מספר הערות:

1. הפירוט הנ"ל מספיק להרצת הסימולטור עם המשקיעים המבוססים על היוריסטיקות אנושיות (המשקיעים מהקובץ inverstors_types/HumanHeuristicsInvestors.py וכן המשקיע שבוחר באקראיות).
2. כדי להריץ את הסימולטור עם פסאודו-משקיעים: BestInvestor ו-WorstInvestor, בנוסף למפורט לעיל יש גם לערוך את הקובץ our_simulator/state.py ולעשות uncomment להערה בשורה 33 (ובכך להכניס לרשימת state_features_list את האיבר 'fund_returns').
3. כדי להריץ את הסימולטור עם אחד המשקיעים הנבונים, RLQInvestor ו-RLAprroximateQInvestor, יש תחילה לייצר את קובץ האימון המתאים כפי שיפורט בהמשך.

תוצאות הרצת הסימולטור עבור המשקיע שנבחר יודפסו למסך.

<u>יצירת קבצי האימון עבור משקיעי Reinforcement Learning</u>

בנינו את הסביבה שלנו כך שיש הפרדה מוחלטת בין שלב האימון של הסוכנים הנבונים (RLQInvestor ו-RLApproximateQInvestor) לבין שלב ההרצה ב-"מצב בחינה". הרצה ב-"מצב בחינה" מצריכה כקלט קבצי pickle שנוצרים במהלך האימון.

אימון הסוכנים מתבצע דרך הקובץ TrainerRL.py, בו יש מחלקה שונה עבור כל סוכן:

- משקיע מסוג RLQInvestor משתמש בקובץ Q-Table.pkl שנוצר לאחר הרצת TrainerQLearning.train().
לאחר הרצה ניתן יהיה למצוא את הקובץ בתיקיית q_learning_q_table.
- משקיע מסוג RLApproximateQInvestor משתמש בקובץ final_weights.pkl שנוצר לאחר הרצת TrainerRL.TrainerApproximateRL.train(). לאחר הרצה ניתן יהיה למצוא את הקובץ בתיקיית approximate_q_learning_weight.

שתי המחלקות הנ"ל דורשות לקבל גם את קובץ ה-csv ושמות הקרנות (דוגמה ב-main בקובץ), וכן ניתן להעביר אליהן גם פרמטרים שונים של האלגוריתמים (לדוגמת γ, ϵ וכו') ואת מספר האפיזודות לאימון.

נשים לב שעבור הרצת הסוכן RLApprximateQInvestor, ישנה אפשרות לתת למאמן שלו כקלט קובץ משקולות קיים – ובכך להמשיך אימון קיים במקום להתחיל מנקודת ההתחלה. במידה ורוצים להמשיך תוצאה של אימון קודם, ניתן לשלוח בפרמטר weights_to_start_dir את הנתיב של קובץ האימון הקודם.

לאחר שהקבצים הנ"ל נוצרו (הסתיים שלב האימון), ניתן להריץ את הסימולטור עם כל אחד מהסוכנים הנ"ל. הסימולטור מחפש בעצמו את קבצי האימון בתיקיות שמצוינות למעלה, אך ניתן לערוך את שמות קבצי האימון (במידה ורוצים להפנות לקבצים אחרים) באמצעות שינוי הערכים המתאימים בתוך הפעולות run_simulator_by_investor (בסימולטור) או run_tests_by_investor (ב-TestLogger שפירוט עליו מופיע למטה).

הרצת ניסויים מרובים וייצוא התוצאות

כעת נסביר כיצד להריץ ניסויים מרובים בבת אחת (כל ניסוי הוא הרצה בודדות עם קבוצה שונה של קרנות אקראיות) וייצוא של התוצאות לקובץ csv לטובת ניתוח התוצאות.

הקובץ הרלוונטי לשלב זה נקרא TestsLogger.py, והוא אחראי להריץ מס׳ כלשהו של ניסויים עבור סוג משקיע קבוע. שלב זה מסתיים ביצירת קובץ logger: קובץ logger הוא קובץ שמכיל את תוצאות של n הרצות של הסימולטור עם משקיע מאותו הסוג. כדי ליצור קובץ כזה יש לבחור את סוג המשקיע מבין המשקיעים השונים בתיקיית investors_types, לבחור את מספר הניסויים n, לעדכן את ה-main של TestsLogger.py בהתאם ולהריץ (לשים לב: ההערות בקשר להרצת הסימולטור במדריך למשתמש תקפות גם לגבי הרצת ה-TestsLogger). לאחר ההרצה, קובץ ה-logger המתאים יופיע בתיקיית experiments_results תחת שם במבנה הבא: [investor_type]_results[date][time].csv.

לדוגמה: כדי ליצור קובץ logger עבור משקיע מסוג BestReturnLastYearInvestor ו-25,000 ניסויים, ה-main של TestsLogger.py צריך להכיל את השורה הבאה:

```
run_tests_by_investor(n=25000, investor_type=BestReturnLastYearInvestor)
```

לאחר מכן ייווצר קובץ logger בשם: ׳BestReturnLastYearInvestor_results 01.05.2021 01-55-33.csv׳.

כל רשומה בקובץ כזה מתעדת ניסוי בודד (הרצה אחת של הסימולטור), כאשר במהלכה אנחנו שומרים את הכסף שהצטבר לאורך כל תור וכן את הבחירה של הסוכן בכל תור.

קבצי התוצאות הספציפיים שנוצרו במהלך הניסויים שלנו, ועל בסיסם ביצענו את הניתוחים, זמינים בגוגל דרייב (קישור בתחתית המסמך)

ניתוח התוצאות

ניתוח התוצאות של הניסויים מבוצעים באמצעות המחברת results_analysis.ipynb. בתא השלישי במחברת יש להזין את הנתיב של התיקייה בה נמצאים קבצי הניסויים (הקבצים שנוצרו בשלב הקודם) לתוך המשתנה working_dir. כל הקבצים צריכים להימצא באותה תיקייה, לדוגמה תיקייה בשם experiments_results שתהיה תחת התיקייה הראשית של הפרויקט.

התא הרביעי במחברת כולל מילון שנכנס למשתנה results_storage. במילון הזה המפתחות הם שמות של סוכנים (שמשתמשים בהם אח״כ בקוד) וה-values הם מילונים נוספים. עבור כל סוכן, ה-value הוא מילון שכולל שם (name) – מחרוזת שתופיעה בכותרת של הגרף, ושם של קובץ (file_name) – השם של קובץ הניסויים שמתאים לסוכן כולל סיומת .csv.

המחברת כבר מלאה בדוגמאות עבור שני השינויים הנדרשים הנ״ל (וכן יש בהערת TODO מה בדיוק צריך לשנות בתחילת כל תא) כדי להריץ את הניתוח עבור קבצי ניסויים ספציפים.

שאר הרצת המחברת מתבצעת באופן סדרתי: שני התאים שלאחר מכן כוללות פונקציות שמייצרות את הגרפים. כל תא לאחר מכן מיועד לסוכן נפרד, וכל שצריך לעשות זה לקרוא לפונקציה full_report_for_signel_heuristic. הפונקציה מצפה לקבל פרמטר אחד, שהוא מילון שכולל את המפתחות name ו-file_name.

כזכור, המילון הנ״ל הוא בדיוק הערך שמתאים למילון results_storage שהוגדר בהתחלה. קריאה לפונקציה לדוגמה: Full_report_for_signel_heuristic(results_storage["LowestFees"]), כאשר LowestFees הוא key במילון results_storage, והערך הוא כאמור מילון שכולל name, file_name.

הפונקציה הנ״ל מדפיסה את הניתוח עבור קובץ הניסויים: מספר סטטיסטיים (מדדי הסיכון, רווח ממוצע וכו׳) וכן את שני הגרפים שהוצגו בדו״ח.

קישורים

את הקבצים שלא יכולנו לצרף להגשה ניתן למצוא בתיקיית גוגל דרייב:

https://drive.google.com/drive/folders/15qAD76yrSGpdRqU6ZLYmoa0_Qz84KSX6?usp=sharing

בנוסף כל הקוד גם זמין ב-GitHub:

https://github.com/roimo19/ai-project-optimal-pension